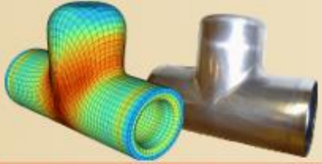


LongLifeAM
Lab Meeting

Liège Université

Maitrise des propriétés de fatigue dans des pièces structurelles produites par fabrication additive
(connaissance, compréhension, prédiction, accroissement)

L.Papeleux

21 december 2020

Timetable

Computing the tenue of AM Piece to fatigue loading through FE^2 Method :

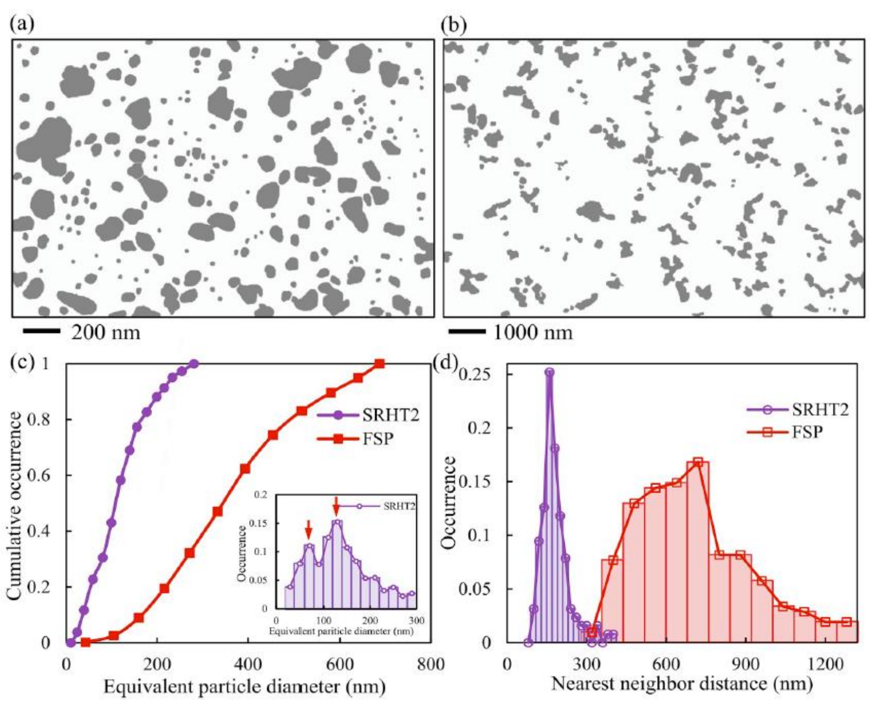
- Microstructure & RVE meshing :
  - Though Python API of GMSH
- New gmshImport : oo_meta\toolbox\gmsh2.py

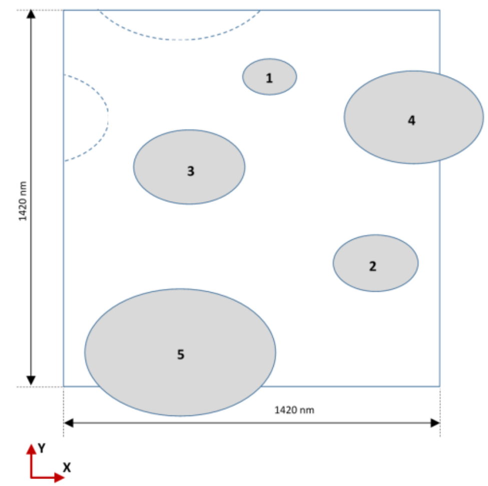
- Applying loads to RVE :
  - New Lagrangian-Multipliers elements to apply linear conditions
  - LM elements to apply repetitivity load to the RVE
  - Validation of the load / results
- Loads of µstructured RVE through LM

VER Meshing

Microstructure & VER meshing :

- µStructure is define by an Al matrix and Si particules (Ellipsoid) in FSP State (pict b)
- First model based on image analysis from UCL (Zhao et al., MSEA 2019)
- Particules sizes and position defined thanks to the analysis of Chantal Bouffioux :





	Interpol.	Ratio=	1.5	centre	centre
	$Ø_{eq}$ partic. (nm)	$Ø_{min}$ (nm)	$Ø_{max}$ (nm)	X (nm)	Y (nm)
1	165.9	135	203	780	1170
2	261.1	213	320	1180	460
3	343.8	281	421	470	830
4	430.5	352	527	1320	1020
5	586.3	479	718	440	130

VER Meshing

Microstructure & VER meshing :

- Model and Mesh built with GMSH (through it Python API)
- All geometrical operations are done through the OpenCascade geometrical engine (not the GMSH builtIn geometry). A priori more easy to use objects.
- 2 mesh generators :
  - 2D / 2.5D (straight extrusion of the 2D Model)
    - 2D - Quadrangle only (when possible it sometimes remains some triangles) and Triangles Meshes
    - 2.5D - Hexa and Penta (also possible to have Tetra)
  - 3D
    - TetraHedron only
- Manage
  - symetrical cuts
  - Rotation of cells
  - Interaction between cells
- Save Mesh in « msh » file (format 2->4)
- Directely interfaced in Metafor tests
- Selections done in gmsh :

VER Meshing

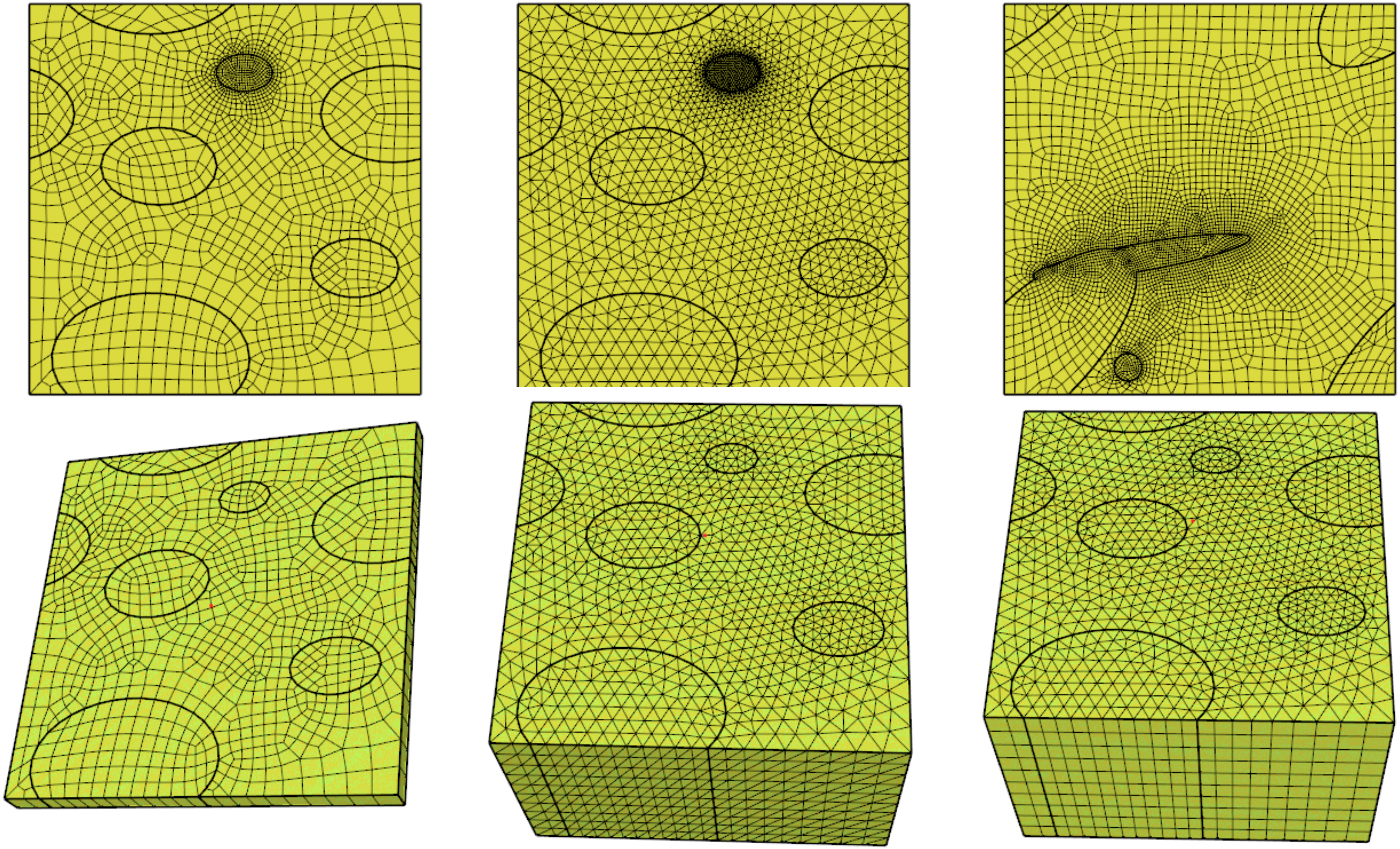
Microstructure & VER meshing :

- 2D / 2.5D (extrusion of the 2D Model)
  - Quadrangle only (even if the doc tells the inverse, it sometimes remains some triangles) and Triangles Meshes
  - Hexahedron (extruded quadrangles) and Pentahedron (extruded triangles) or Tetrahedron

```
p['ver'] = t.Rectangle(X0, Y0, Lx, Ly, cl)
p['cells']=[]
p['cells'].append(t.Circle(id, sym, X, Y, R, cl))
p['cells'].append(t. Ellipse(id, sym, X, Y, R1, R2, alpha, cl))
… # max 90  cells
p['nz']   = 1 # nz != 0 active l’extrusion 3D
p['thick']   = 100.0e-3
p['recombine2D']    = True #quad
#p['recombine2D']    = False # Tri
p['recombine3D'] = True # penta  ou hexa selon optition recombine2D
#p['recombine3D'] = False #Tetra necessite  rec2D = False
```

VER Meshing

Microstructure & VER meshing :



VER Meshing

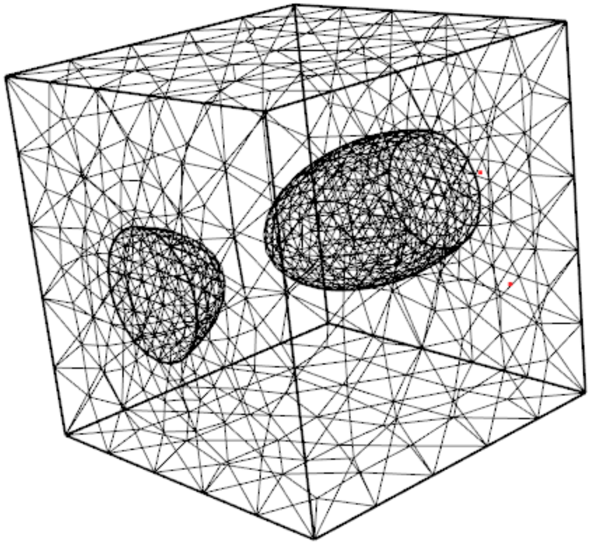
Microstructure & VER meshing :

- 3D : Tetraedron only

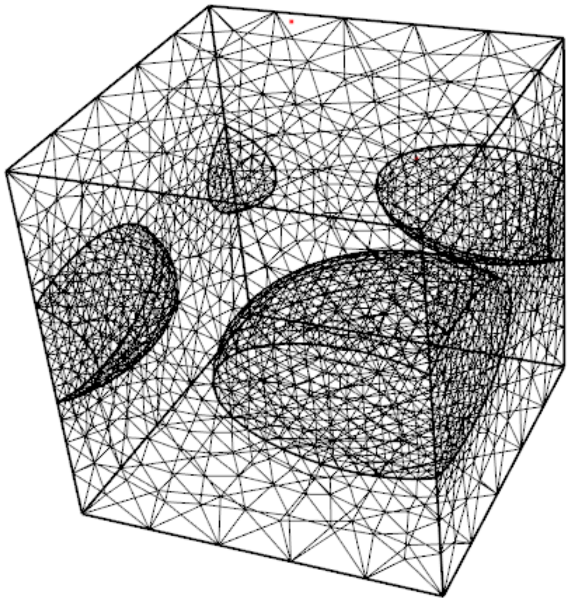
```
p['ver'] = t.HexaBox(X0, Y0, Z0, Lx, Ly, Lz, cl)
p['cells']=[]
p['cells'].append(t.Sphere(id, sym, X, Y, Z, R, cl))
p['cells'].append(t.Ellipsoid(id, sym, X, Y, Z, R1, R2, R3, alphaX, alphaY, alphaZ, cl))
 … # max 90  cells
p['recombine2D']    = True #quad
#p['recombine2D']    = False # Tri
p['recombine3D'] = True # penta  ou hexa selon optition recombine2D
#p['recombine3D'] = False #Tetra necessite  rec2D = False
```

VER Meshing

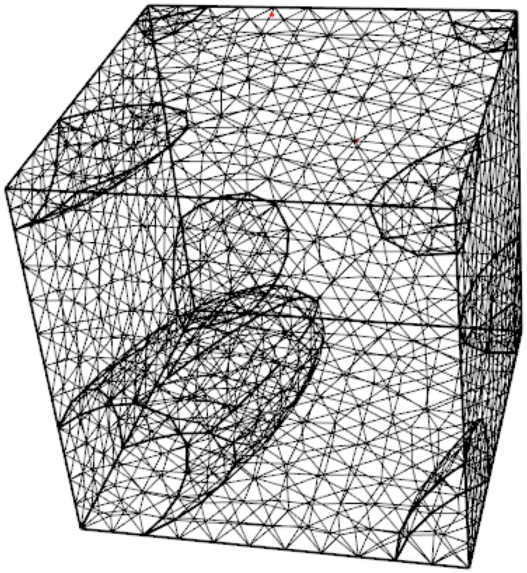
Microstructure & VER meshing :



Side Intersection



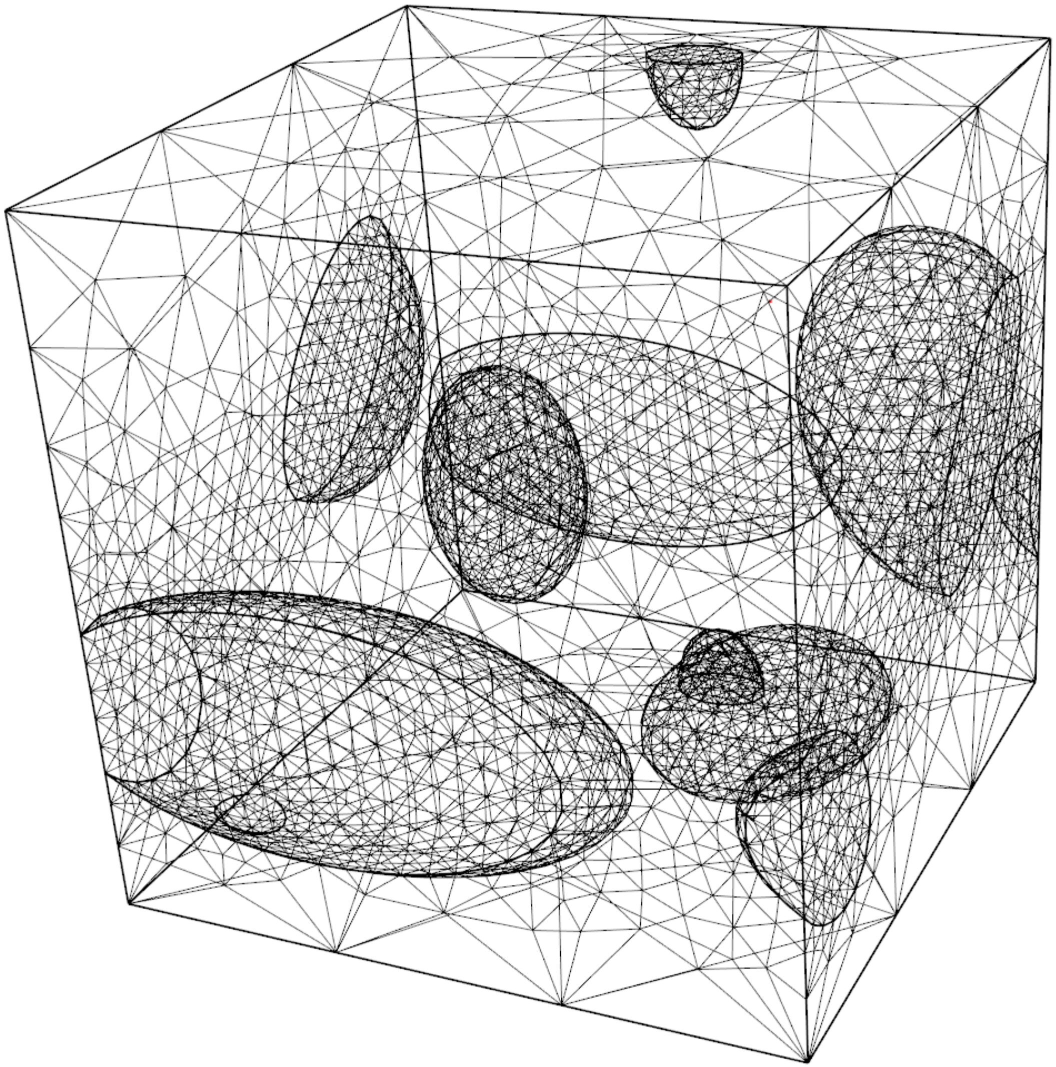
Edge Intersection



Corner Intersection

VER Meshing

Microstructure & VER meshing :



VER Meshing

Microstructure & VER meshing :

- Selections done in gmsh (imported in Metafor):

Physical id:

Group no : 1 id : bgMaterial filled with : 1409 nodes
Group no : 2 id : cellsMaterial filled with : 496 nodes

Group no : 11 id : cell1 filled with : 215 nodes
Group no : 12 id : cell1symX filled with : 52 nodes
Group no : 13 id : cell1symY filled with : 61 nodes
Group no : 14 id : cell1symZ filled with : 69 nodes
Group no : 15 id : cell1symXY filled with : 30 nodes
Group no : 16 id : cell1symYZ filled with : 18 nodes
Group no : 17 id : cell1symZX filled with : 33 nodes
Group no : 18 id : cell1symXYZ filled with : 18 nodes

Group no : 101 id : bgBnd filled with : 977 nodes
Group no : 102 id : cellsBnd filled with : 461 nodes
Group no : 111 id : cell1Bnd filled with : 180 nodes
Group no : 112 id : cell1symXBnd filled with : 52 nodes
Group no : 113 id : cell1symYBnd filled with : 61 nodes
Group no : 114 id : cell1symZBnd filled with : 69 nodes
Group no : 115 id : cell1symXYBnd filled with : 30 nodes
Group no : 116 id : cell1symYZBnd filled with : 18 nodes
Group no : 117 id : cell1symZXBnd filled with : 33 nodes
Group no : 118 id : cell1symXYZBnd filled with : 18 nodes

Group no : 203 id : bgCellsBnd filled with : 293 nodes
Group no : 211 id : bgCell1Bnd filled with : 131 nodes
Group no : 212 id : bgCell1symXBnd filled with : 25 nodes
Group no : 213 id : bgCell1symYBnd filled with : 36 nodes
Group no : 214 id : bgCell1symZBnd filled with : 44 nodes
Group no : 215 id : bgCell1symXYBnd filled with : 17 nodes
Group no : 216 id : bgCell1symYZBnd filled with : 9 nodes
Group no : 217 id : bgCell1symZXBnd filled with : 20 nodes
Group no : 218 id : bgCell1symXYZBnd filled with : 11 nodes

VER Meshing

Microstructure & VER meshing :

- Selections done in gmsh (imported in Metafor):

VER Topology

Group no : 1001 id : ptX0Y0Z0 filled with : 1 nodes
Group no : 1002 id : ptX1Y0Z0 filled with : 1 nodes
Group no : 1003 id : ptX1Y1Z0 filled with : 1 nodes
Group no : 1004 id : ptX0Y1Z0 filled with : 1 nodes
Group no : 1005 id : ptX0Y0Z1 filled with : 1 nodes
Group no : 1006 id : ptX1Y0Z1 filled with : 1 nodes
Group no : 1007 id : ptX1Y1Z1 filled with : 1 nodes
Group no : 1008 id : ptX0Y1Z1 filled with : 1 nodes
Group no : 1011 id : lineY0Z0 filled with : 12 nodes
Group no : 1012 id : lineX1Z0 filled with : 13 nodes
Group no : 1013 id : lineY1Z0 filled with : 12 nodes
Group no : 1014 id : lineX0Z0 filled with : 13 nodes
Group no : 1015 id : lineY0Z1 filled with : 12 nodes
Group no : 1016 id : lineX1Z1 filled with : 13 nodes
Group no : 1017 id : lineY1Z1 filled with : 12 nodes
Group no : 1018 id : lineX0Z1 filled with : 13 nodes
Group no : 1021 id : lineX0Y0 filled with : 13 nodes
Group no : 1022 id : lineX1Y0 filled with : 13 nodes
Group no : 1023 id : lineX1Y1 filled with : 13 nodes
Group no : 1024 id : lineX0Y1 filled with : 13 nodes
Group no : 1031 id : sideX0 filled with : 182 nodes
Group no : 1032 id : sideX1 filled with : 182 nodes
Group no : 1033 id : sideY0 filled with : 185 nodes
Group no : 1034 id : sideY1 filled with : 185 nodes
Group no : 1035 id : sideZ0 filled with : 187 nodes
Group no : 1036 id : sideZ1 filled with : 187 nodes

VER Meshing

Microstructure & VER meshing :

- Bugs – Todo
  - OCC Bounding Boxes may return false values (=> workaround through option ''OCCBoundsUseStl')
  - OCC bug in intersect command (workaround through option 'useIntersectByFragments')

```
# occ method of computation of the boundingBox
p['OCCBoundsUseStl'] = 0 # 0 : may induce a absolute (1.0e-7) extension of the bounding box
                  #(used by default with a not too strict tolBBox)
#p['OCCBoundsUseStl'] = 1 # 1 : is theorically more accurate but may fail in small size geometries
p['tolBBox'] = 1.0e-4
p['useIntersectByFragments'] = True # try to recover the buggy occ intersect through segment cmd
#p['useIntersectByFragments'] = False # use the occ intersect cmd
```

  - More efficient management of cell length
  - More easy to use rotation of ellipsoid (up to now : RZ->RY->RX)
  - Still bug remaining in imposing Periodicity of the mesh (3D) => without gmsh error (detail check of failing tests before alerting CG)
  - Defining interface cell/matrice elements (I have to discuss the format with Chantal)
  - Not limit to straight extrusions
  - Too small interface (cut size ~0) => periodicity problems

GMSH Import

- Previous version outdated (only available to import gmsh files format msh2)
- New file format very different…
- Need to go through a written file
- Does not import names of the Physical Entities

- => New gmsh import (oo_meta\toolbox\gmsh2.py – in my branch)
- Usage compatible (Initial values not imported and command lines options has to be defines in mesh definition)
- Interface .geo, .msh (any readable version), python scripts, any model readable by gmsh,…
- Though the PythonAPI (=> will remain independant to msh file format)
- Read the gmsh DB in memory and directly defines objects in the Metafor DB
- Return a dictionnary of Physical entities in the tests
- All (but1) tests of the battery allready migrated (in my branch)
- Need to install the gmsh SDK version (and to add gmsh in PythonPath)
- Only Elements of dim of space are imported (choice)
- « Integration » to Metafor ⇔ Licence question

GMSH Import

- Limits :
  - Initial values are not imported
  - Arguments of gmsh not imported (does not work) => all options has to be defined in .geo (exepted the mesh command that is automatically called if mesh is not generated)
  - ShiftPointFct : not tested (no tests in the battey)

```
# import du fichier .geo
from toolbox.gmsh import GmshImport
f = os.path.join(os.path.dirname(__file__), './gmshtest.geo')
mesher = GmshImport(f, domain)
~~mesher.setOpti(True)~~   # Must be defined in .geo
~~mesher.setOrder(2)~~     # Must be defined in .geo
mesher.execute()
```

→

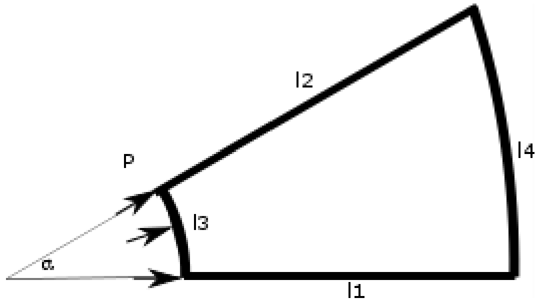
```
# import du fichier .geo
from toolbox.gmsh2 import GmshImport
f = os.path.join(os.path.dirname(__file__), 'gmshtest.geo')
importer = GmshImport(f, domain)
importer.verb = True
importer.writeMshFile = True
importer.mshFileFormat = 2.0
importer.setNodeIdx(0)
importer.execute()

groups = importer.groups
```

LM to apply Linear Conditions Constrains :

- New Lagrangian-Multipliers elements to apply linear conditions
  - Cell type : CELL_LM
  - Number of Nodes : 2->5 (not necessary different)
  - ElementProperties :
    - LinearConstraintLagMult2/3DElement
    - Properties :
      - FIELD1 -> FIELD5 now : Displacement Fields : Field1D(TX/TY/TZ, AB/RE)
      - FACTOR1->FACTOR5
  - Enforce the linear equation (through Lagrange Multipliers):
      - FACTOR1*Node1.FIELD1 + FACTOR2*Node2.FIELD2 + FACTOR3*Node3.FIELD3 + FACTOR4*Node4.FIELD4 + FACTOR5*Node5.FIELD5=0
  - Need the use of a solver allowing nul pivoting (DSSolver does, Skyline does not, Mumps?)
  - Analytical Stiffness available
  - Compute internal Forces (allows to extract values for VER). Other LM elements compute ExternalForces. We may keep the 2 kinds of elements if needed…

Applying loads to VER :

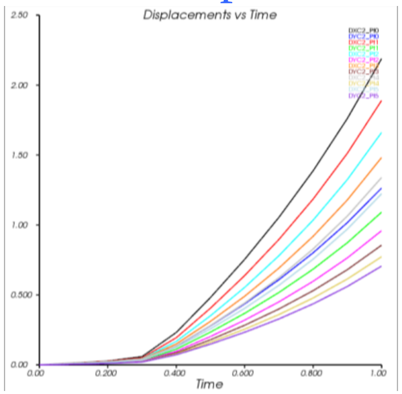
- Exemple : Radial Constraint : Thick Cylinder submitted to internal pressure
  - Contraints : Nodes of Line 2 has to move radially
    - $dX = dR\,cos(\alpha)$
    - $dY = dR\,sin(\alpha)$
    - $\Rightarrow dR = \frac{dX}{cos(\alpha)} = \frac{dY}{sin(\alpha)} \Rightarrow \frac{dX}{cos(\alpha)} - \frac{dY}{sin(\alpha)} = 0$

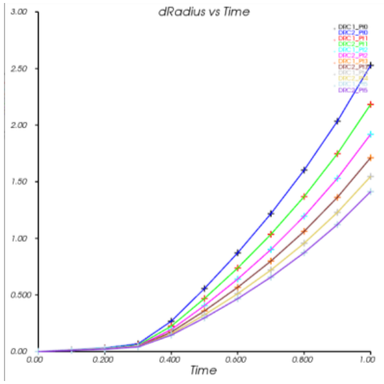


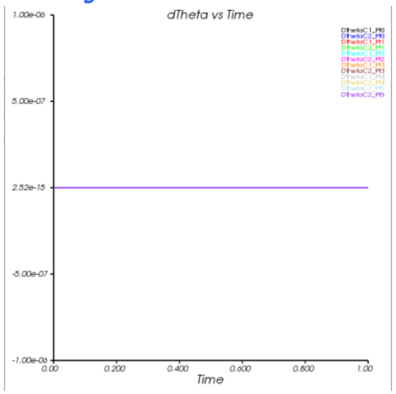

```
mesh = geometry.getMesh()
num = 1000
g1001 = gset.add(Group(1001))
for i in range(0,l2.getNumberOfMeshPoints()):
    num += 1
    mesh.define(num, CELL_LM, g1001, [l2.getMeshPoint(i),l2.getMeshPoint(i)])
prp1001 = ElementProperties(LinearConstraintLagMult2DElement)
prp1001.put(FIELD1, Field1D(TX, RE))
prp1001.put(FACTOR1, 1.0/cosA)
prp1001.put(FIELD2, Field1D(TY, RE))
prp1001.put(FACTOR2, -1.0/sinA)
lmi = LagMultInteraction(1001)
lmi.push(g1001)
lmi.addProperty(prp1001)
interactionset.add(lmi)
```

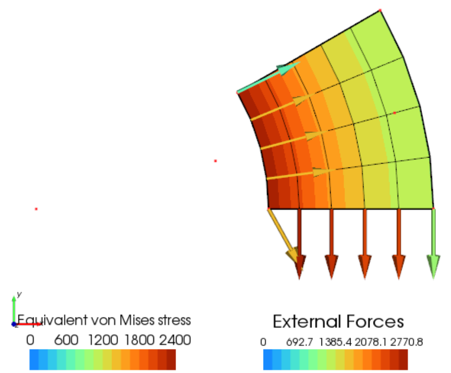
LM constrains:

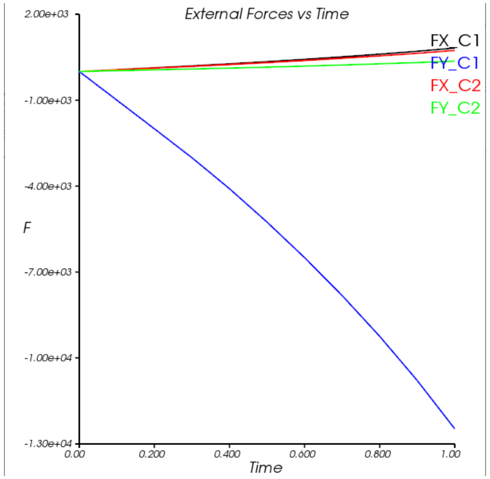
- Exemple : Radial Constraint : Thick Cylinder submitted to internal pressure

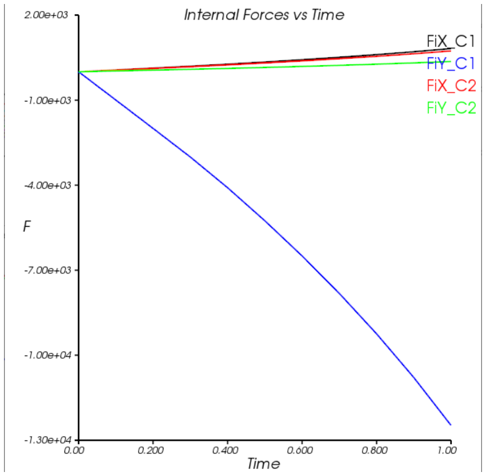


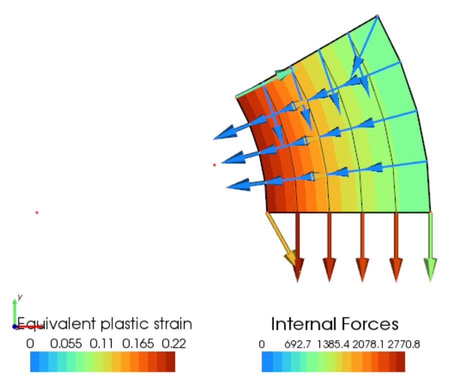








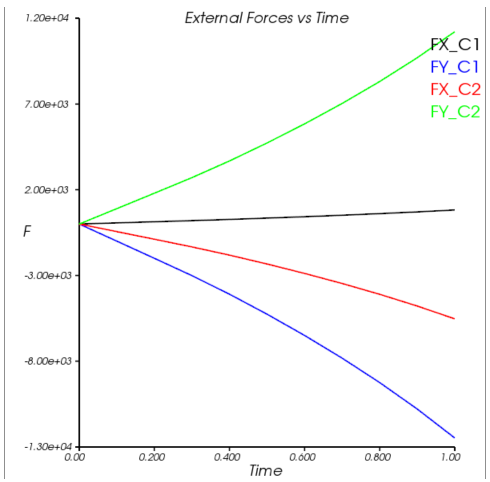


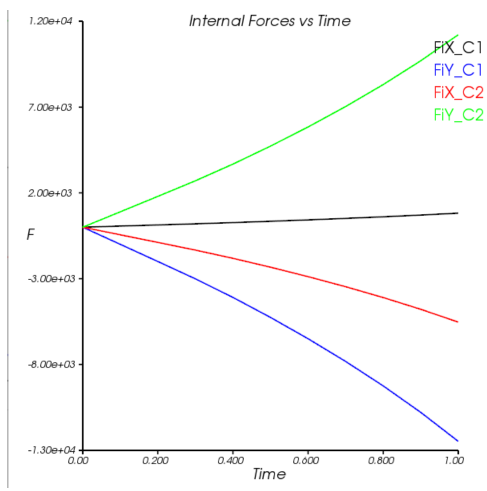


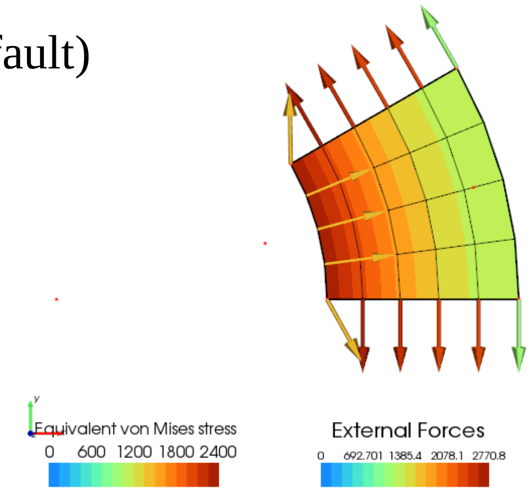
- Radial constraints perfectly respected
- Impossible to measure reaction on nodes where constraints are applied

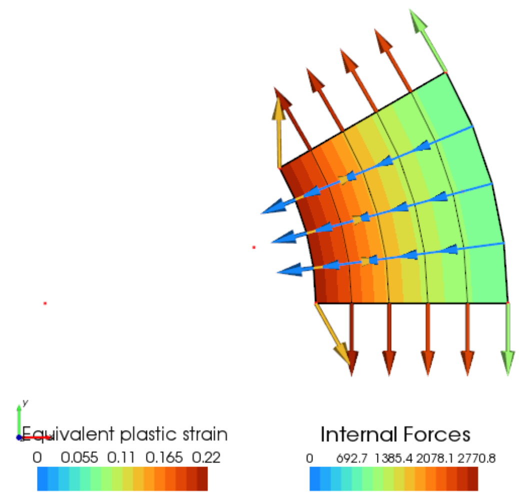
LM constrains:

- Exemple : Radial Constraint : Thick Cylinder submitted to internal pressure
  - Alternative : Compute LM condition on FExt instead of Fint
  - Boolean Propelem Option : COMPUTEFEXT (False by default)
    - Reaction forces are measured on constrained nodes



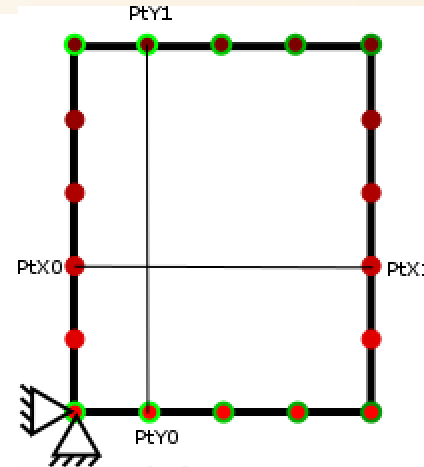




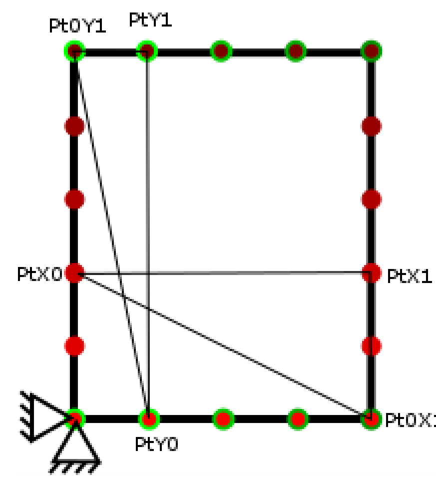


- Total Load : FY_C1 = -12474.5 / FTheta_C2 = 12501.3
- Si on maille + finement selon theta :
  - FY_C1 = -12532.4 / FTheta_C2 = 12533.5
- Condition :
  - All nodes of the interaction are on the same radial line (while condition is the same for all points)

Applying loads to RVE : Periodicity conditions

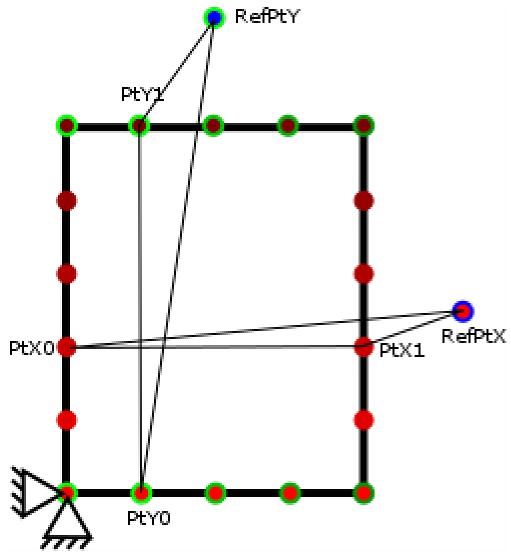


- Goal : keep the possibility to assemble Vers to each others
  - Initial mesh has to respect periodicity conditions
  - Displacements of nodes has to respect periodicity
    - dX(PtX0) = dX(PtX1)
    - dY(PtX0) = dY(PtX1)
    - ....
  - Upper condition is too strict (does not allows poisson effect for example)
  - After discussion with Chantal : Definition of conditions with respect to a refence node whose displacement reflect the global strain:
    - dX(PtX1)-dX(PtX0) = dX(Pt0X1)
    - dY(PtX1)-dY(PtX0) = dY(Pt0X1)
    - ...



  - Loads are applied to all nodes (depending of the load)
  - Discussions with Chantal how to load correctly the structure (without too many boundaries effect).
    The idea was to duplicate a part of the RVE Mesh
  - => loading is complicated (may conflict with each other on corners nodes)

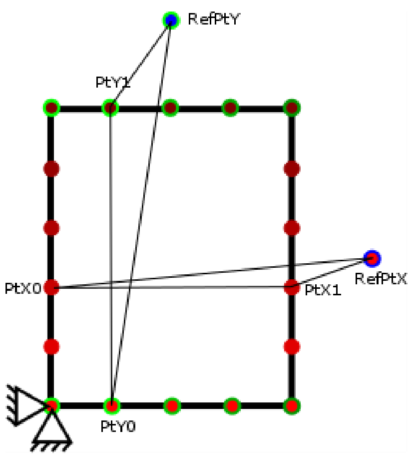
Applying loads to RVE : Periodicity conditions

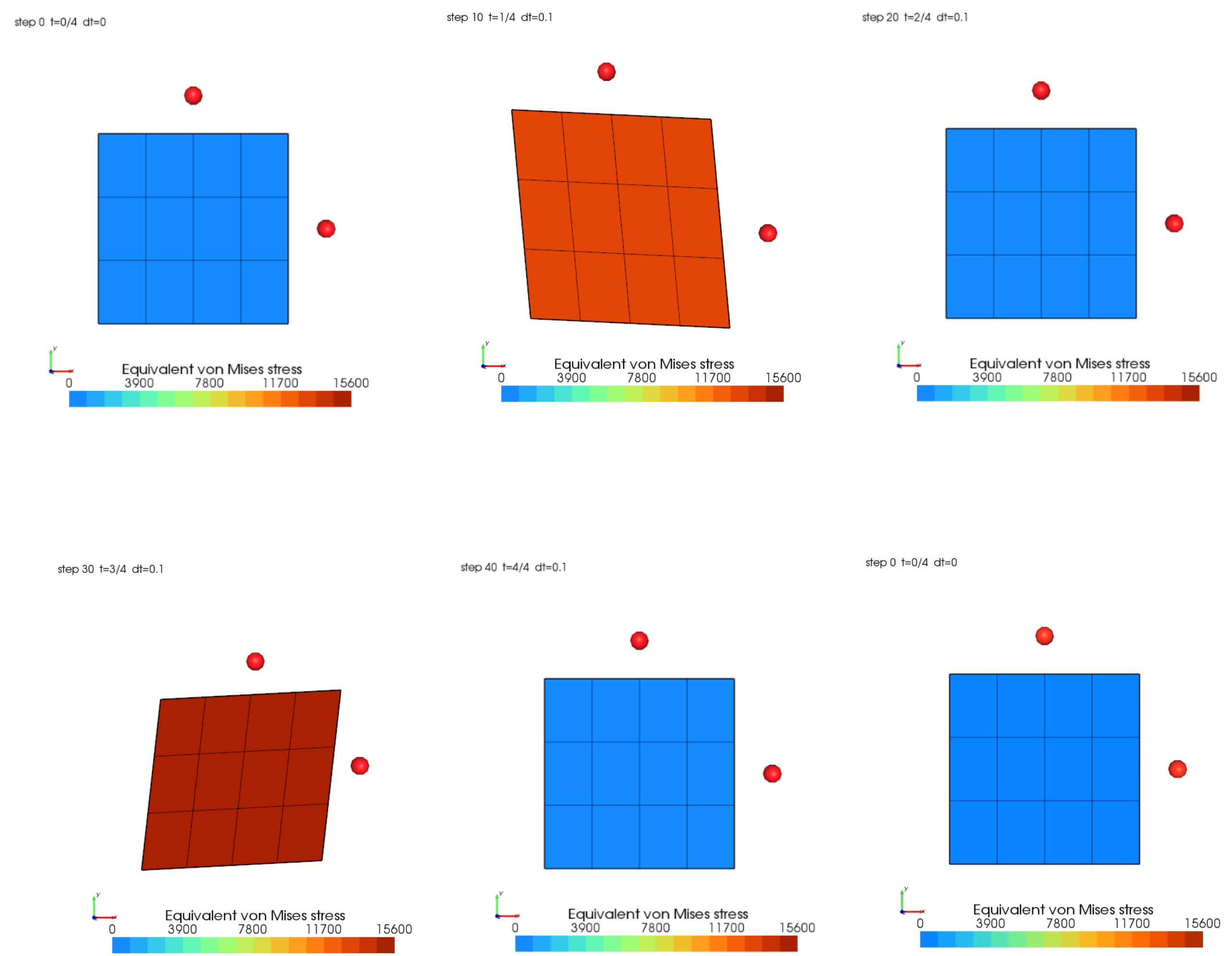


- Project MOCOMAC :
  - In Digimat (Marc), RVE are defined with external Refence Points (RefPtX and RefPtY)
  - The periodicity conditions are defined by :
    - dX(PtX1)-dX(PtX0) = dX(RefPtX)
    - dY(PtX1)-dY(PtX0) = dY(RefPtX)
    - …
  - The loadings / fixations are applied to RefPoints
  - It is possible to let a side free keeping the repetitivity condition (just unfix the corresponding refPt displacement)
  - For FE^2 :
    - Applying a strain tensor is directely linked to displacement of the corresponding RefPt
    - External Forces measured on RefPt will directly gives the stress of the Macro scale

Applying loads to RVE : Periodicity conditions

Example : ver2D – Plane Strain – Elastic Material

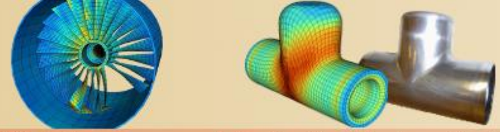




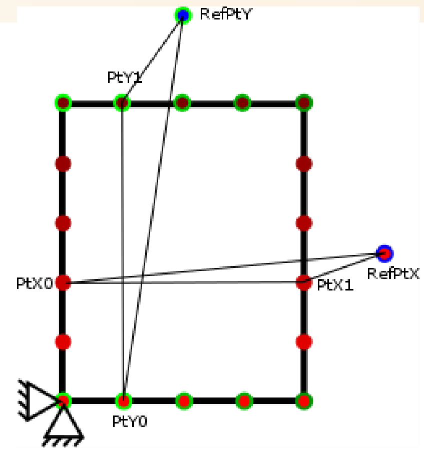
Load Applied :

- dX refPtX = 0.05
- dY refPtX = -0.05
- dX refPtY = -0.1
- dY refPtY = 0.1

- Fix Rigid Modes (Translation)
  - dX(Pt1) = dY(Pt1)=0



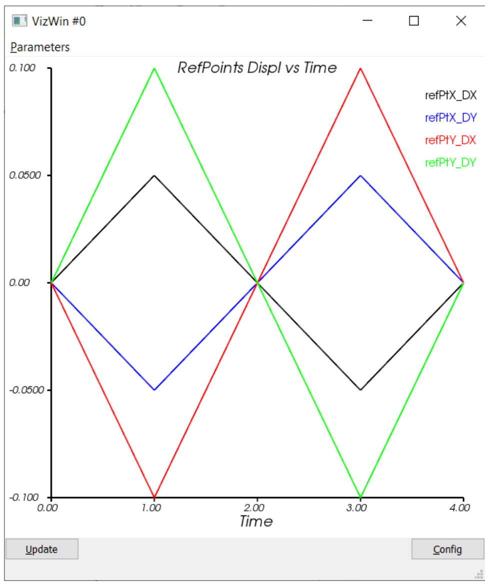
Applying loads to RVE : Periodicity conditions

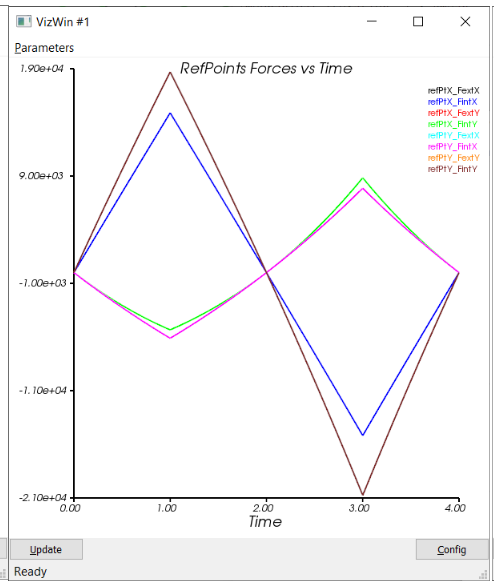


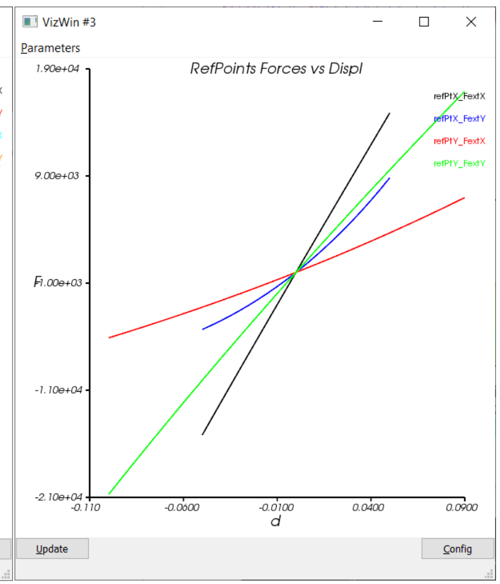
Example : ver2D – Plane Strain – Elastic Material

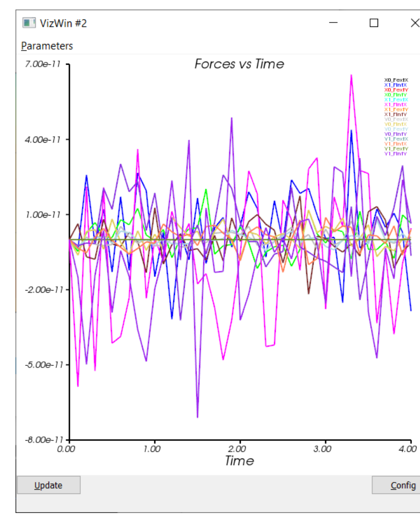
Load Applied :

- dX refPtX = 0.05
- dY refPtX = -0.05
- dX refPtY = -0.1
- dY refPtY = 0.1





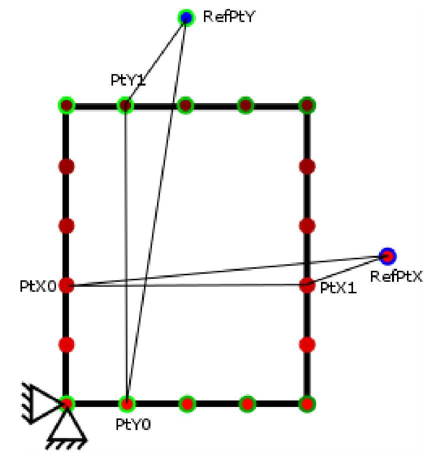




[TSC-STP] Number of steps : 40
[TSC-ITE] Number of mech. iterations : 43

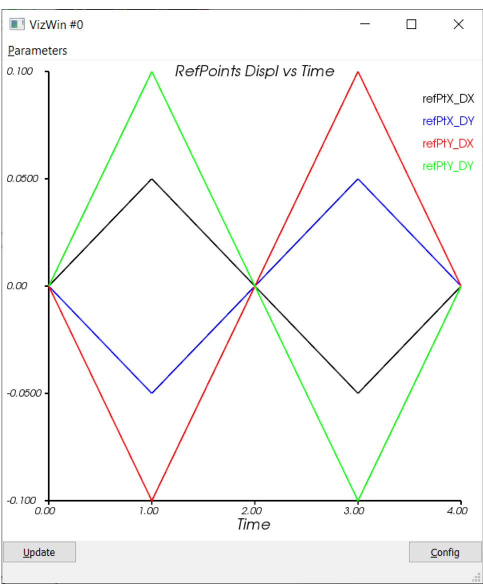
Applying loads to RVE : Periodicity conditions

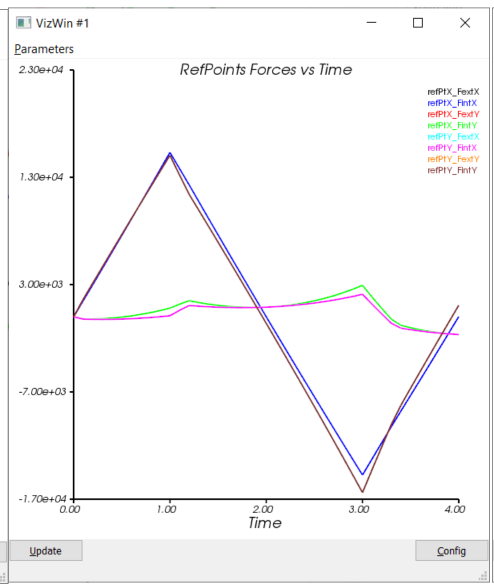
Example :ver2D Plane Strain / Elasto-Plastic Mterial

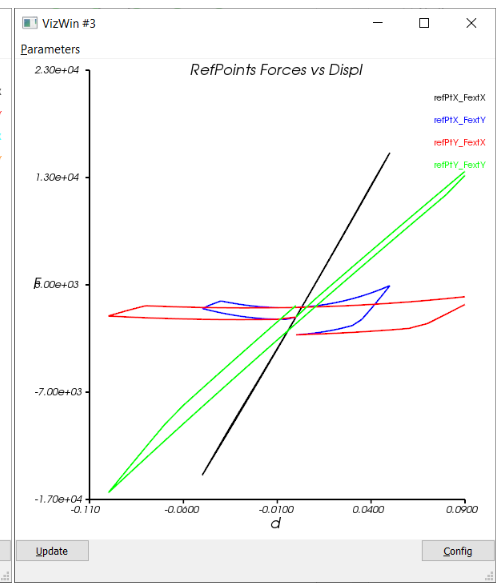


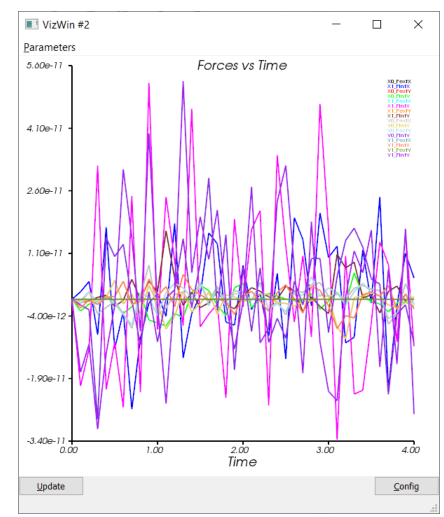
Load Applied :

- dX refPtX = 0.05
- dY refPtX = -0.05
- dX refPtY = -0.1
- dY refPtY = 0.1

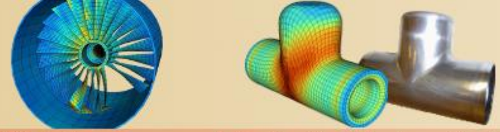








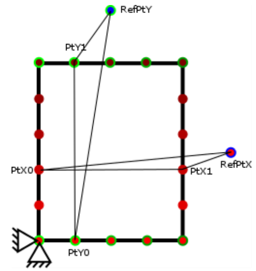
[TSC-STP] Number of steps : 40
[TSC-ITE] Number of mech. iterations : 43



Applying loads to RVE : Compare to Mono-Element

Traction simple Elastique

RVE :

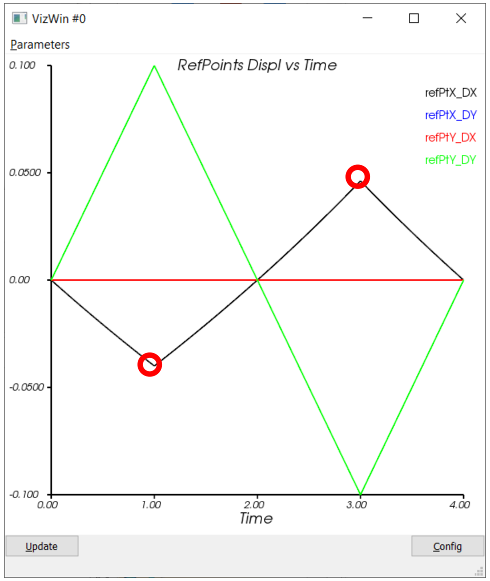


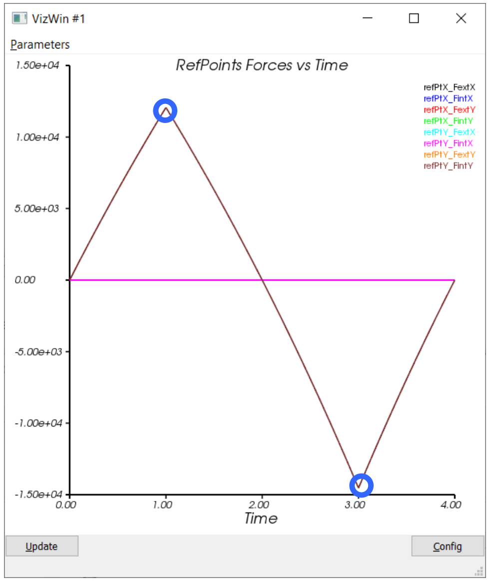
Load Applied :

- dX refPtX = Free
- dY refPtX = Free
- dX refPtY = Free
- dY refPtY = 0.1

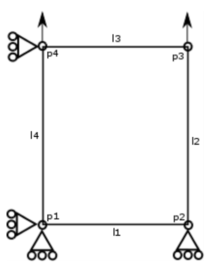
dX_refPtX(1) = -4.0024217359474368e-02
dX_refPtX(3) = 4.6189487885506975e-02

FY_refPtY(1) = 1.2065335793695258e+04
FY_refPtY(3) = -1.4535437392632848e+04





Mono-Element

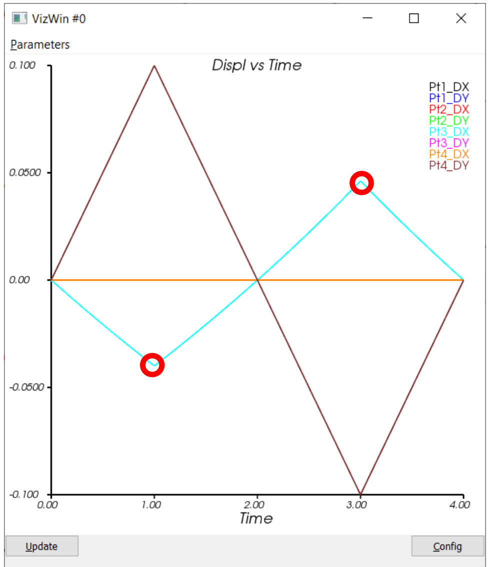


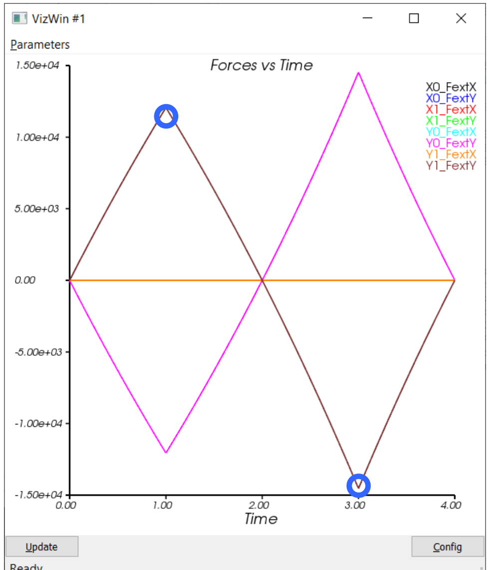
Load Applied :

- dX l4 = 0.0
- dY l1 = 0.0
- dY l3 = 0.1

dX_p3(1) = -4.0024217359474319e-02
dX_p3(3) = 4.6189487885506898e-02

FY_l3(1) = 1.2065335882643809e+04
FY_l3(3) = -1.4535437196502364e+04

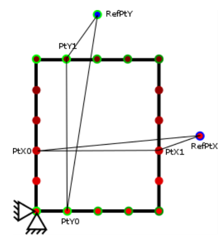




Applying loads to RVE : Periodicity conditions

- Traction simple Elasto-Plastique

RVE :

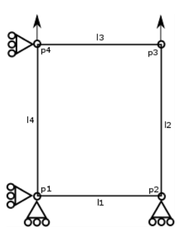


Load Applied :

- dX refPtX = Free
- dY refPtX = Free
- dX refPtY = Free
- dY refPtY = 0.1

Dx_refPtX(1) = -8.4279345496135152e-02
Dx_refPtX(3) = 9.2464828187837120e-02
Dx_refPtX(4) =
FY_refPtY(1) = 1.3432722833012713e+03
FY_refPtY (3) = -3.7325907898913938e+03
FY_refPtY (4) = 4.0591141646801739e+03

Mono-Element

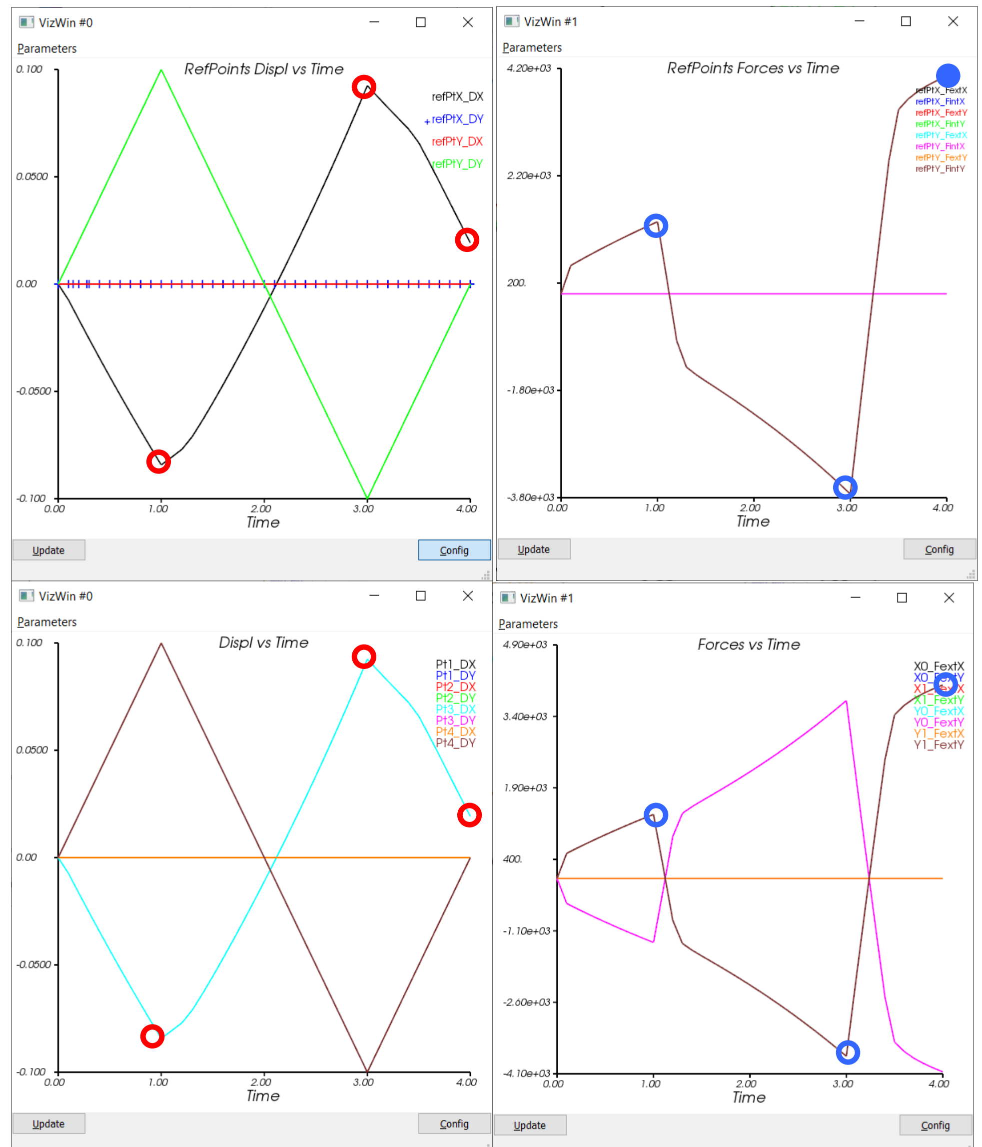


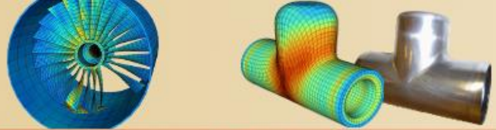
Load Applied :

- dX l4 = 0.0
- dY l1 = 0.0
- dY l3 = 0.1

dX_p3(1) = -8.4279397508710163e-02
dX_p3(3) = 9.2464861479245250e-02
dX_p3(4) =

FY_l3(1) = 1.3432652787129059e+03
FY_l3(3) = -3.7325841794000453e+03
FY_l3(4) = 4.0591091971531359e+03





Applying loads to µStructure RVE

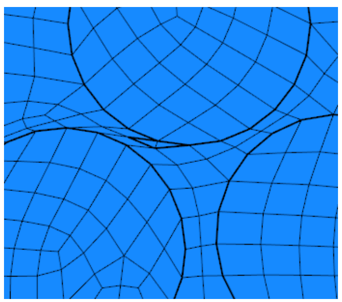
Load :

p['dxx'] = None *#0.0*

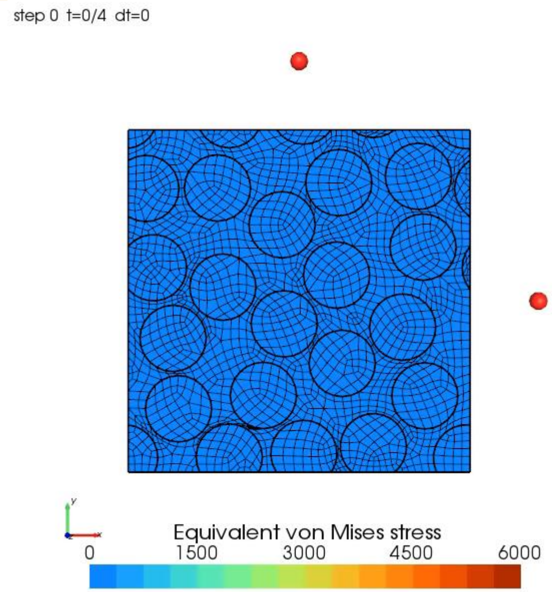
p['dxy'] = None *#0.0*

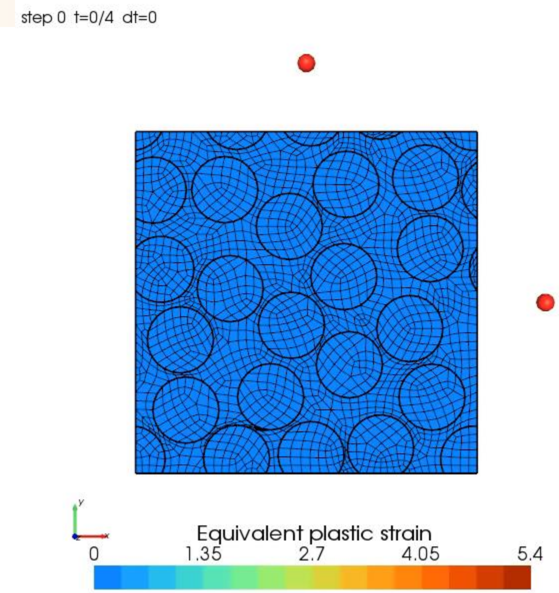
p['dyx'] = 0.0

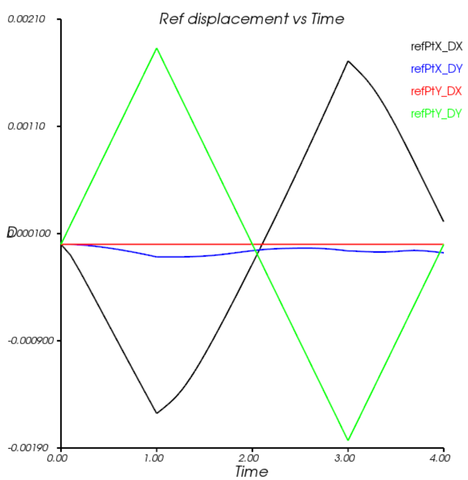
p['dyy'] = 36.635e-3/20.0

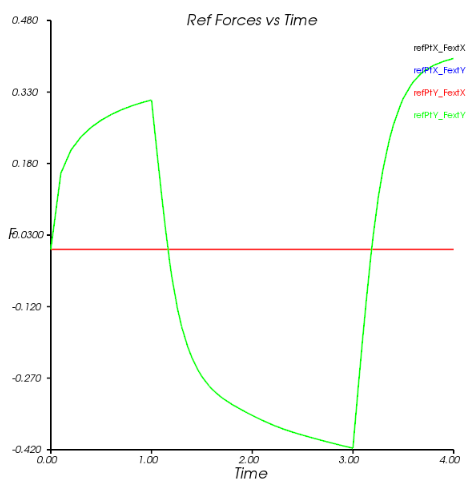


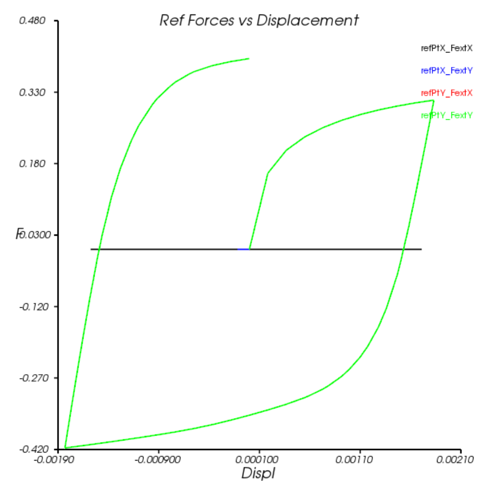
Remaining triangle







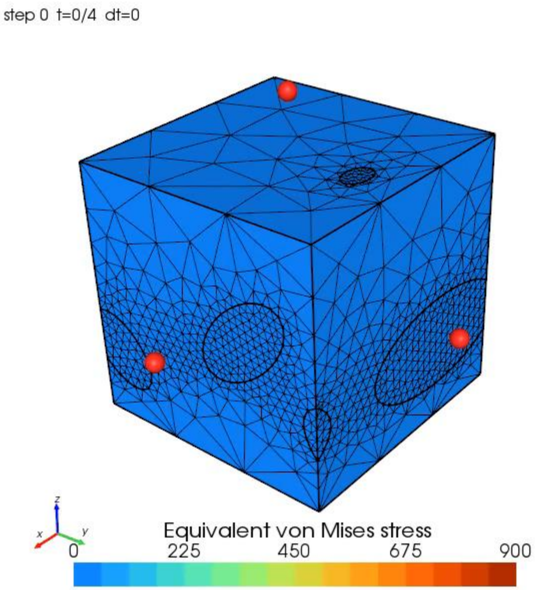


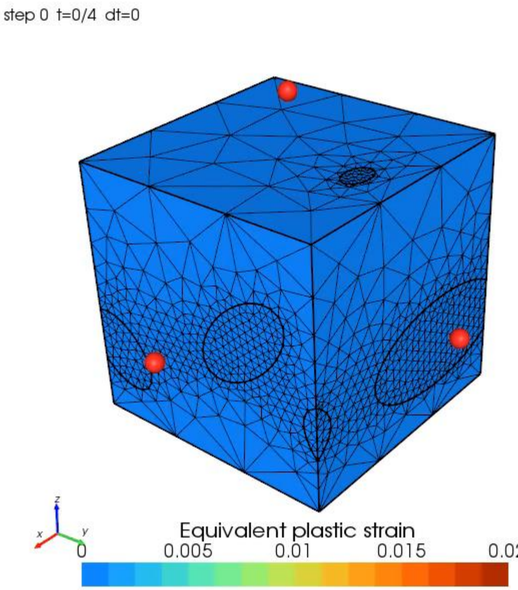


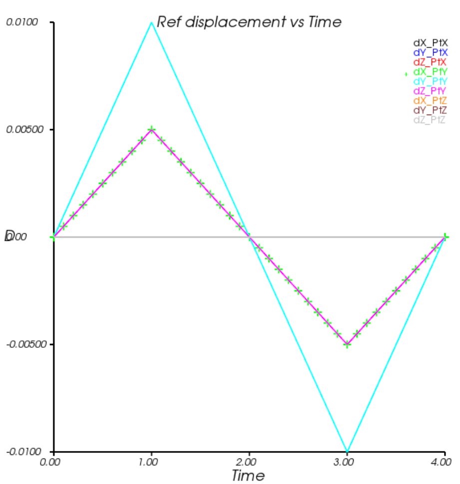
Applying loads to µStructure RVE

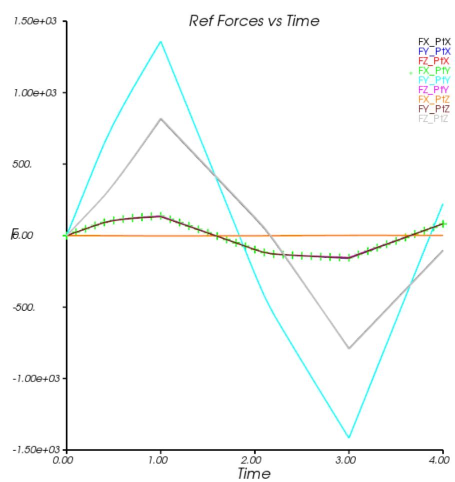
Load :

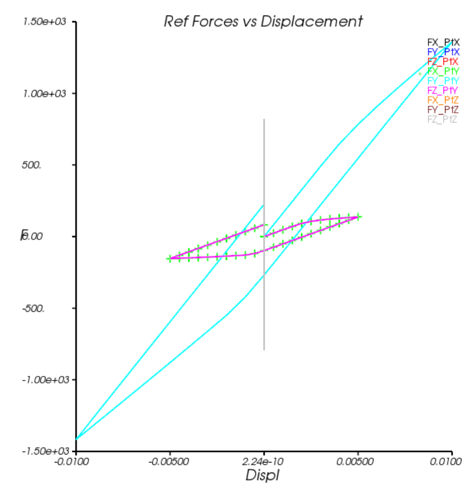
- p['dxx'] = 0.0
- p['dxy'] = 0.0
- p['dxz'] = 0.0
- p['dyx'] = 10.0e-3 / 2.0
- p['dyy'] = 10.0e-3
- p['dyz'] = 10.0e-3 / 2.0
- p['dzx'] = 0.0
- p['dzy'] = 0.0
- p['dzz'] = 0.0











Applying loads to RVE : Periodicity conditions

- Todo :
  - Check some details on the model
    - Force/displ curves in 3D
    - Clean outputs (add verb where needed)
    - Bug loadFac (does not loads the curves anymore : may be linked to MR !15)
  - Commit all changes & MR
  - Computation :
    - Natural Strains -> references NodesDispl
    - References Nodes Forces -> Cauchy Stresses
      (is this necessary to compute precisely the length (area) of the edges or is straigth line enough?)
  - Visualisation :
    - LM cells ? (how)
    - vectors on « refPt »

Applying loads to RVE : Periodicity conditions

- Biblio :
  - C. Geuzaine and J.-F. Remacle. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, 2009